Private, but restricted, access to databases

J. Herranz[1]

***Abstract*—We consider the primitive of *restricted oblivious transfer*: the owner of a database wants to restrict the access of users to this data according to some policy, whereas a legitimate user wants to privately retrieve allowed parts of the data, without letting the owner know which part of the data is being obtained.**

We formally describe the protocols and required properties for restricted oblivious transfer, and we explain how the techniques of *priced oblivious transfer* [1] can be used to solve the problem in the particular case where the family of restrictions is a weighted threshold one. We state a conjecture on a new characterization of these kind of families. Then we propose a generic solution for the problem of restricted oblivious transfer, which works for any possible monotone family of restrictions. The solution is right now of theoretical interest only, because it uses a cryptographic tool which has not been realized yet: cryptosystems which are both multiplicatively and additively homomorphic.

***Index Terms*—Oblivious transfer, weighted threshold families, homomorphic encryption.**

I. Introduction

With the growth of the Internet, many practical situations involving operations on digital confidential data appear constantly. On the one hand, the owner of the data wants it to remain private against those users who do not have the right to access it. This can be achieved by keeping the data in a secure device, or by encrypting it. Many examples of such confidential databases can be found in the areas of medical analysis, electronic commerce, banking, or digital business in general. On the other hand, some users can have the right to access to some parts of this confidential data, because of their role (doctors) or because they pay for it. In some situations, these allowed users may want to keep private what part of the data they are retrieving. Imagine the following motivating example inspired by *pay per view* systems: a TV channel over the Internet broadcasts different programs (films, sport events...) which can be watched only by those clients who have paid for them. A client may pay a registration fee which gives him the right of watching five films and ten football matches in the current month, for example. A solution for the TV channel is to keep a private database containing one password for each program, and then to allow the download of the program only to those users having the corresponding password. On the one hand, if a client has paid only to watch films, he should have access only to the part of the database which contains the passwords for the films. On the other hand, when asking for a password to watch a film, maybe the client does not want the TV channel to know which film he is going to watch.

Summing up, there are situations where both the owner of the data (from now on, the *server*) and the users who have access to some parts of the data (from now on, the *clients*) want to preserve some kind of privacy. This problem of *restricted oblivious transfer* is the one that we consider in this work. The cryptographic primitive of standard oblivious transfer [9], [4] ensures the privacy for the clients. It is an interactive protocol between a server and a client: the client retrieves an item db_i, and nothing else, from a database $DB = \{db_1, \ldots, db_N\}$ of secret items maintained by the server, who does not obtain any information about the index i (chosen by the client) of the retrieved item. Of course, such a protocol does not solve, by itself, the problem of restricted oblivious transfer: since the server does not know the index of the queried item, he cannot decide if the client has the right to obtain this item or not.

Although the problem of restricted oblivious transfer is, in our opinion, very interesting and of great importance in a society demanding more and more privacy, it has not received a lot of attention. The only (directly) related work is the paper by Aiello et al. [1], where they consider and realize the primitive of *priced oblivious transfer*: each item of the database has a price, each client has a budget, and the client can privately retrieve/buy items as long as his money balance (which is updated by the server at the end of each execution) remains positive.

In this paper we first describe in detail the general primitive of restricted oblivious transfer: how to model the restrictions of clients, by using (decreasingly) monotone families of subsets; which are the inputs and outputs of each of the protocols; and which are the required security properties for these protocols. Once this is done, we explain how priced oblivious transfer can be recovered as a particular case of this primitive, when the family of restrictions is a *weighted threshold* one. However, priced oblivious transfer is far from covering all cases of restricted oblivious transfer, because many montone families are not weighted threshold; to emphasize this last statement, we state a conjecture on a new and simple characterization of weighted threshold families.

Then we present a first solution to the general problem of restricted oblivious transfer. It is inspired by the solution presented in [1] to the problem of priced oblivious transfer. However, differently than in the priced case, the general solution that we propose makes use of a very special cryptographic tool: public key encryption schemes which are at the same time additively and multiplicatively homomorphic. Since such schemes are not known to exist yet, the interest of our solution is only theoretical, right now.

a) Organization of the paper.: In Section II we recall the concepts of homomorphic encryption, conditional disclosure of a value, and (weighted threshold) monotone families. In Section III we detail the primitive that we want to implement, and we explain a particular cases of it: priced oblivious transfer. We discuss our conjecture on a new characterization of

[1]IIIA-CSIC, Bellaterra, Spain. jherranz@iiia.csic.es

weighted threshold families in Section IV. Then, we propose our solution to the general problem of restricted oblivious transfer, of theoretical interest only, in Section V. We conclude our work in Section VI.

II. Preliminaries

In this section we recall three concepts which will appear in the rest of the work.

A. Homomorphic Encryption

A public key encryption scheme $PKE = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ consists of three probabilistic and polynomial time algorithms. The key generation algorithm $\mathcal{KG}$ takes as input a security parameter (for example, the desired length for the secret key) and outputs a pair (sk, pk) of secret and public keys. The encryption algorithm takes as input a plaintext m and a public key pk, and outputs a ciphertext $c = \mathcal{E}_{pk}(m)$. Finally, the decryption algorithm takes as input a ciphertext and a secret key, and gives a plaintext $m = \mathcal{D}_{sk}(c)$ as output.

Such a scheme has an *homomorphic* property if there exist two operations, defined on the set of ciphertexts and plaintexts, respectively, such that the result of operating two ciphertexts is an encryption of the result of operating the two corresponding plaintexts. For example, a public key cryptosystem is *additively* homomorphic if there exists an operation $\oplus$ defined on the set of ciphertexts, such that the message encrypted in $c_1 \oplus c_2$ is $m_1 + m_2$, where m_i is the message encrypted in c_i, for $i = 1, 2$. Formally, this property is written as

$$\mathcal{D}_{sk}\big(\mathcal{E}_{pk}(m_1) \oplus \mathcal{E}_{pk}(m_2)\big) \;=\; m_1 + m_2.$$

Analogously, a cryptosystem is *multiplicatively* homomorphic if there exists an operation $\otimes$ defined on the set of ciphertexts, such that $\mathcal{D}_{sk}\big(\mathcal{E}_{pk}(m_1) \otimes \mathcal{E}_{pk}(m_1)\big) = m_1 \cdot m_2$, for any pair of plaintexts (m_1, m_2). Of course, these definitions make sense only if the multiplication and addition operations are properly defined on the set of plaintexts.

ElGamal cryptosystem [5] is the classical example of multiplicatively homomorphic scheme, whereas Paillier's one [8] is an additively homomorphic encryption scheme. Boneh et al. [3] proposed a public key encryption scheme which is, at the same time, additively and multiplicatively homomorphic; but the multiplicatively homomorphic property is satisfied only if the product operation is applied once. Homomorphic cryptosystems have a lot of applications, including electronic auctions and electronic voting.

B. Conditional Disclosure of Secrets

The general *conditional disclosure* primitive was introduced in [6]. In this work we are interested in the following particular case: a server holds a secret value s, a message m and a public key pk corresponding to some (additively homomorphic) cryptosystem PKE. A user holds the matching secret key sk, and a message m'. By using the conditional disclosure primitive, the user gives $\mathcal{E}_{pk}(m')$ to the server, and he must obtain secret s from the server if and only if the condition $m' = m$ is satisfied. The server must obtain no information about m'.

This primitive can be realized in the following way: after receiving $\mathcal{E}_{pk}(m')$, the server takes at random an element γ and, using the homomorphic property of PKE, computes an encryption c of $\rho = \gamma(m - m') + s$. The server sends c to the user, who can use sk to decrypt and obtain β, which is equal to the secret s if $m = m'$, and is a random value otherwise.

This simple protocol solves the conditional disclosure problem in the case of a single equality between a value known to the server and an encrypted value. The more general case where the condition is a monotone formula with equality leaves can be similarly solved, using the following recursive method, as proposed in [1]. On the one hand, to realize the conditional disclosure of secret s under condition $C_1 \vee C_2$, one can run two independent instances of the protocol, one under condition C_1 and the other under condition C_2. On the other hand, to realize the conditional disclosure of secret s under condition $C_1 \wedge C_2$, one can choose r at random and run two independent instances of the conditional disclosure protocol, one for secret r under condition C_1, and the other for secret $s + r$ under condition C_2.

C. Monotone Families of Subsets

Given a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of n elements, and a family $\mathcal{B} \subset 2^{\mathcal{P}}$ of subsets of $\mathcal{P}$, we say that $\mathcal{B}$ is (decreasingly) *monotone* if $B_1 \in \mathcal{B}$ and $B_2 \subset B_1$ imply that $B_2 \in \mathcal{B}$. This kind of families appear very often in real life applications: any family $\mathcal{B}$ which contains those subsets of people who are not authorized to "do something" must be decreasingly monotone.

Because of the monotone property, any monotone family $\mathcal{B}$ is completely defined by its *basis* $\mathcal{B}_0$, which contains the maximal subsets of $\mathcal{B}$. That is,

$$\mathcal{B}_0 = \{B \in \mathcal{B} \mid \forall p_i \notin B,\ B \cup \{p_i\} \notin \mathcal{B}\}.$$

Analogously, given a decreasingly monotone family $\mathcal{B}$, we can consider its complementary family $\bar{\mathcal{B}} = \{A \mid A \notin \mathcal{B}\}$, which is obviously an increasingly monotone family. It is therefore defined by its basis $(\bar{\mathcal{B}})_0$ of minimal subsets:

$$(\bar{\mathcal{B}})_0 = \{A \notin \mathcal{B} \mid \forall p_i \in A,\ A - \{p_i\} \in \mathcal{B}\}.$$

A particular case of (decreasingly) monotone families are *weighted threshold* ones. A family $\mathcal{B}$ is weighted threshold if there exist an assignment $\omega : \mathcal{P} \to \mathbb{Z}^+$ of positive integers, and a positive threshold $\beta \in \mathbb{Z}^+$ such that

$$B \in \mathcal{B} \iff \sum_{p_i \in B} \omega(p_i) \leq \beta.$$

III. Restricted Oblivious Transfer

In this section we explain in more detail the general functionality of restricted oblivious transfer (phases, inputs/outputs of the protocols, desired security properties). We then review some particular cases of this functionality, which can be realized quite directly by using existing results.

A. Definitions: Functionality and Properties

Let us remember the functionality we want to implement: a server S maintains a secret database $DB = \{db_1, \ldots, db_N\}$ with N entries/items, and a policy defining which subsets of entries of the database can be available, on request, to the different clients. A client wants his requests to be private, i.e. the server should not obtain any information about the item(s) that the client is requesting. The server wants to be sure that a client C will not obtain any information about items of the database which are not allowed to C. In general, a solution fulfilling this functionality will consist of two protocols:

- **1st protocol: defining rights.** This phase should be run off-line, maybe before the specific values of the entries $\{db_i\}_{1 \leq i \leq N}$ of the database are defined. Let $\mathcal{I} = \{1, 2, \ldots, N\}$ denote the set of indices of the items in the database. For a particular client C, the server S specifies the family $\mathcal{B}_C \subset 2^{\mathcal{I}}$ of subsets of items that client C is allowed to obtain. Of course, $\mathcal{B}_C$ must be a decreasingly monotone family: if $B_1 \in \mathcal{B}_C$ is allowed, and $B_2 \subset B_1$, then $B_2 \in \mathcal{B}_C$ is allowed, as well. The family $\mathcal{B}_C$ is known by both S and C. The server S stores an information $\texttt{info}_C$ related to C, which initially contains $\mathcal{B}_C$ and which is updated by S each time C requests an item of the database. Possibly, the client C receives some additional information α_C from S, to be used in the future requests.
- **2nd protocol: request and retrieval.** The input for the client C includes α_C and the index i corresponding to the entry db_i he wants to retrieve from the database. The input for the server consists of the database DB and $\texttt{info}_C$. At the end of the protocol, S must update his information $\texttt{info}_C$, and C obtains a value $\texttt{out}_i$. Assume that this is the t-th time that C executes this protocol with S, and that previous executions had inputs $i_1, \ldots, i_{t-1}$. Let us define the subset of indices $B = \{i_1, \ldots, i_{t-1}, i\}$. Then C obtains the desired value, i.e. $\texttt{out}_i = db_i$, if and only if $B \in \mathcal{B}_C$.

A scheme for this functionality of restricted oblivious transfer will be considered valid if it satisfies some requirements. Assume that the t-th execution of the protocol has input (i_t, α_C) for C, where $i_t \in \{1, \ldots, N\}$. The first requirement is a typical correctness one: if the client and the server behave honestly during the t-th execution and if $\{i_1, \ldots, i_t\} \in \mathcal{B}_C$, then $\texttt{out}_{i_t} = db_{i_t}$ is the secret output of C. Additionally, two privacy properties are required.

- **Privacy for the client.** In any execution of the protocol for request and retrieval of an item, the server S does not obtain any information about the index i in the input of the client.
- **Privacy for the server.** In the t-th execution of the 'request and retrieval' protocol, with input (i_t, α_C), the client C
  - does not obtain any information about items db_ℓ, for $\ell \neq i_t$; and
  - does not obtain any information about item db_{i_t}, if $\{i_1, \ldots, i_t\} \notin \mathcal{B}_C$.

B. A Particular Case: Priced Oblivious Transfer

When the family $\mathcal{B}_C \subset 2^{\mathcal{I}}$ is a weighted threshold family, which means that there exist a threshold β and an assignment of positive weights $\omega : \mathcal{I} \to \mathbb{Z}^+$ such that

$$\mathcal{B}_C = \{B \subset \mathcal{I} \text{ s.t. } \sum_{i \in B} \omega(i) \leq \beta\},$$

then we recover *priced oblivious transfer* [1], where elements of the database are supposed to be objects (or goods) with a price, and users have different budgets to privately buy goods. In the solution given in [1], the prices of the objects are the same for all the users/buyers. But their solution can be applied in general to the restricted oblivious transfer problem where each user can have different restrictions (not only the budget), as long as all the restriction families $\mathcal{B}_C$ are weighted threshold.

Namely, assume that the restrictions of a client are described by a weighted threshold family $\mathcal{B}_C$, which is realized by a threshold β and an assignment of weights $\omega : \mathcal{I} \to \mathbb{Z}^+$. All this information is known by both the client and the server. We present a brief sketch of how the basic solution proposed in [1] would work.

- All the weights are assumed to be different (if necessary, by scaling them and the threshold with a large enough factor): $\omega(i) \neq \omega(j)$, if $i \neq j$.
- The client generates a pair of secret/public keys (sk, pk) for an additively homomorphic cryptosystem; encryption is denoted as $c = \mathcal{E}_{pk}(m)$.
- The server encrypts $c = \mathcal{E}_{pk}(\beta)$.
- If the client wants to retrieve element db_i, he sends $\tilde{c} = \mathcal{E}_{pk}(\omega(i))$ to the server, along with a (cryptographic) proof that the value encrypted in c is greater or equal than the value encrypted in $\tilde{c}$. The client adds a standard oblivious transfer query for the index i.
- The server verifies that the validity proof is correct. If so, he answers the standard oblivious transfer query, and updates $c = c \ominus \tilde{c} = \mathcal{E}_{pk}(\beta - \omega(i))$, for possible future executions of the protocol with the same client.
- The client recovers db_i using the last step of the oblivious transfer protocol, and updates $\beta = \beta - \omega(i)$ and $c = c \ominus \tilde{c}$.

This solution forces the server to store a lot of different information (the threshold β, the assignment of weights ω, the ciphertext c) for each client. This drawback, as well as some of the techniques used in the priced oblivious transfer protocols of [1], will appear in the generic solution to the problem of restricted oblivious transfer that we present in next section.

IV. Characterizing Weighted Threshold Families?

In order to see that priced oblivious transfer is quite far from solving all the cases of restricted oblivious transfer, we want to show that there are many decreasingly monotone families which are not weighted threshold. Recall that a decreasingly monotone family $\mathcal{B} \subset 2^{\mathcal{P}}$, where $\mathcal{P} = \{p_1, \ldots, p_n\}$, is weighted threshold if there exist a positive integer β and an assignment of weights $\omega : \mathcal{P} \to \mathbb{Z}^+$ such that $B \in \mathcal{B}$ if and only if $\sum_{p_i \in B} \omega(p_i) \leq \beta$.

Notation. Given an assignment of weights $\omega : \mathcal{P} \to \mathbb{Z}^+$, we will use the following notation, for simplicity: $\omega_i = \omega(p_i)$, for any element $p_i \in \mathcal{P}$; and $\omega_B = \sum_{p_i \in B} \omega(p_i)$, for any subset $B \subset \mathcal{P}$.

For example, consider the set $\mathcal{P} = \{p_1, p_2, p_3, p_4\}$ and the family $\mathcal{B}$ defined by its basis $\mathcal{B}_0 = \{\{p_1, p_2\}, \{p_2, p_3\}, \{p_3, p_4\}\}$. This family cannot be weighted threshold; in effect, if suitable ω and β existed, then we would have $\omega_2 < \omega_3$, on the one hand, because $\{p_1, p_2\} \in \mathcal{B}$ and $\{p_1, p_3\} \notin \mathcal{B}$. On the other hand, since $\{p_3, p_4\} \in \mathcal{B}$ and $\{p_2, p_4\} \notin \mathcal{B}$, we would have $\omega_3 < \omega_2$; a contradiction.

Our conjecture is that the existence of four such subsets R, S, U, Y (in the example, $R = \{p_1\}, S = \{p_4\}, U = \{p_2\}, Y = \{p_3\}$) is a sufficient and necessary condition to ensure that a family is not weighted threshold.

Conjecture 1: A decreasingly monotone family $\mathcal{B} \subset 2^{\mathcal{P}}$ is a weighted threshold one if, and only if, there do not exist subsets $R, S, U, Y \subset \mathcal{P}$ such that

(i) $R \cap U = S \cap Y = \emptyset$;
(ii) $R \cup U \in \mathcal{B}$, $R \cup Y \notin \mathcal{B}$;
(iii) $S \cup U \notin \mathcal{B}$, $S \cup Y \in \mathcal{B}$.

There is an implication, necessity, which is easy to prove. Before doing this, however, we have to note that the given condition on R, S, U, Y is equivalent to a similar one, which is obtained by adding an additional requirement (iv) $R \cap Y = S \cap U = \emptyset$. In effect, if there exist R, S, U, Y satisfying (i), (ii) and (iii) in the conjecture, then we can define $R' = R - (R \cap Y)$ and $S' = S - (S \cap U)$, and then the tuple (R', S', U, Y) satisfies (i)-(iv).

Necessity

Assume that $\mathcal{B}$ is a weighted threshold family realized by an assignment $\omega : \mathcal{P} \to \mathbb{Z}^+$ and a threshold $\beta \in \mathbb{Z}^+$. Assume then, to the contrary, that there exist subsets $R, S, U, Y \subset \mathcal{P}$ satisfying (i)-(iii) or, equivalently, satisfying (i)-(iv). On the one hand, $R \cup U \in \mathcal{B}$ and $R \cup Y \notin \mathcal{B}$, combined with $R \cap U = R \cap Y = \emptyset$, imply $\omega_U < \omega_Y$. On the other hand, $S \cup U \notin \mathcal{B}$ and $S \cup Y \in \mathcal{B}$, combined with $S \cap U = S \cap Y = \emptyset$, imply $\omega_U > \omega_Y$. Therefore, we would have a contradiction; we conclude that such subsets R, S, U, Y cannot exist.

Sufficiency

Right now, we have not been able either to prove or to disprove the sufficiency implication in general. For the particular case of families $\mathcal{B}$ of rank 2 (where subsets in $\mathcal{B}$ have at most two elements), however, we have been able to prove our conjecture, by proving that our condition on R, S, U, Y is equivalent to a known characterization of rank 2 weighted threshold families, given in [7]

We think that it is worth to study our conjecture in detail, if possible by validating it with a formal proof. The resulting characterization of weighted theshold families would be very interesting, due to its simplicity. In particular, it would lead to simpler results for some particular families (bipartite families, rank 2 families), for example in the area of secret sharing [10], [2].

V. A Generic Solution for Restricted Oblivious Transfer

The previous section shows that there are many monotone families which are not weighted threshold ones. Therefore, if the family $\mathcal{B}_C$ of restrictions of a client to access a database is such a non-weighted threshold family, the ideas of priced oblivious transfer cannot be used to implement restricted oblivious transfer.

We propose in this section a first generic solution which works for any possible family $\mathcal{B}_C$ of restrictions. It can be seen as a generalization of the protocols in [1] for priced oblivious transfer. Unfortunately, the solution is, at the current time, of theoretical interest only, because it employs a cryptographic tool which has not been realized yet: cryptosystems which are both multiplicatively and additively homomorphic. Let us assume, anyway, the existence of such a cryptosystem, $PKE = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$, with operations $\otimes$ and $\oplus$ defined on the set of ciphertexts, satisfying for any two plaintexts m_1, m_2:

$$\mathcal{D}_{sk}\big(\mathcal{E}_{pk}(m_1) \otimes \mathcal{E}_{pk}(m_2)\big) = m_1 \cdot m_2, \text{ and}$$
$$\mathcal{D}_{sk}\big(\mathcal{E}_{pk}(m_1) \oplus \mathcal{E}_{pk}(m_2)\big) = m_1 + m_2.$$

The two protocols of the generic solution that we propose work as follows.

b) 1st protocol: defining rights.: Let $\mathcal{B}_C = \{B_1, B_2, \ldots, B_s\} \subset 2^{\mathcal{I}}$ be the family of subsets of indices expressing the collections of items that client C is allowed to query, where $\mathcal{I} = \{1, 2, \ldots, N\}$. In general, we will have $B_j = \{i_{j,1}, i_{j,2}, \ldots, i_{j,n_j}\} \subset \mathcal{I}$. We represent each of these subsets B_j with its *incidence vector*, a length N binary vector $\vec{b}_j = (b_1^{(j)}, b_2^{(j)}, \ldots, b_N^{(j)}) \in \{0,1\}^N$, such that $b_i^{(j)} = 1$ if and only if $i \in B_j$, for $i = 1, 2, \ldots, N$. For simplicity of notation, we will sometimes skip the super-index $^{(j)}$, when it is clear from the context.

The client generates a pair of keys $(pk, sk) \leftarrow \mathcal{KG}(1^k)$ for the homomorphic cryptosystem PKE, and publishes pk. The server computes an encryption of $\mathcal{B}_C$, by encrypting all the vectors $\vec{b}_j$ coordinate-wise. In other words, he computes $\vec{c}_j = (\mathcal{E}_{pk}(b_1^{(j)}), \ldots, \mathcal{E}_{pk}(b_N^{(j)}))$, for $j = 1, \ldots, s$. He also chooses a value u_0. Initially, he sets $\texttt{info}_C = (pk, u_0, \{\vec{c}_j\}_{j=1,\ldots,s})$.

c) 2nd protocol: request and retrieval.: We assume that db_i are non-negative integer values. If the client wants to retrieve the item db_i, then he must encrypt the vector $\vec{e} = (e_1, \ldots, e_N) \in \{0,1\}^N$, which has a 1 in the i-th position, $e_i = 1$, and a 0 everywhere else, $e_\ell = 0$, $\forall \ell \neq i$. The result is a vector of ciphertexts $\vec{c} = (c_1, c_2, \ldots, c_N) = \big(\ \mathcal{E}_{pk}(e_1), \ldots, \mathcal{E}_{pk}(e_N)\ \big)$. If the client has behaved honestly, then the value

$$\tilde{c} = \bigoplus_{\ell=1}^{N} (db_\ell) c_\ell$$

is an encryption of db_i, due to the (additively) homomorphic properties of the scheme PKE. Here $(db_\ell)c_\ell$ means aplying db_ℓ times the operation $c_\ell \oplus c_\ell$. Now the server has to update the access family $\mathcal{B}_C$, because the i-th item has been already queried and will be released. For each subset $B_j \in \mathcal{B}_C$, the server updates its representation vector $\vec{b}_j$, by replacing a 1 with a 0 in the i-th position, if $b_i^{(j)} = 1$ (which means that

index i belonged to B_j) and by setting all the vector $\vec{b}_j$ to 0, if $b_i^{(j)} = 0$ (which means that the client has asked for an item which was not in B_j, and so the subset B_j must not be considered any more). This is done via the formula

$$\vec{b'}_j = [\vec{b}_j \cdot \vec{e}](\vec{b}_j \ - \vec{e}),$$

where $\vec{b'}_j$ denotes the updated version of the vector associated to subset B_j. The server must perform this operation over encrypted data, $\vec{c}'_j = [\vec{c}_j \odot \vec{c}](\vec{c}_j \ - \vec{c})$; this is the point where we need PKE to be both multiplicatively and additively homomorphic. Furthermore, we will have to perform two levels of products of ciphertexts, one for the 'scalar' product $\vec{c}_j \odot \vec{c}$, and another one to multiply the resulting ciphertext with each component of the vector of ciphertexts $(\vec{c}_j \ - \vec{c})$. For this reason, the scheme in [3], which allows only one level of products of ciphertexts, cannot be used here.

Of course, we have to consider the possibility of dishonest clients. The server will release to the client this value $\tilde{c}$ (an encryption of db_i), conditioned to the fact that the query is *correct*, which means that:

1) the client has behaved honestly in the previous executions of the protocol; and
2) the vector of ciphertexts $\vec{c} = (c_1, \ldots, c_N)$ contains exactly one encryption of 1, and $N-1$ encryptions of 0; and
3) the index i such that c_i is an encryption of 1 still belongs to some subset $B_j \in \mathcal{B}_C$.

The first condition is ensured with a chaining technique [1]: in the t-th execution of the protocol, the client receives an encryption of some value u_t, if and only if the t-th query is correct. Then, in the $(t+1)$-th execution, the client must provide an encryption of u_t.

The second condition ensures that the client will obtain at most one item, if any, and that the client cannot try to "increase" his rights $\mathcal{B}_C$ for the following interaction, by defining $e_\ell = -1$ for some $\ell \in \{1, \ldots, N\}$, for example. The server will not know which entry of $\vec{c}$ encrypts the value 1, but he will be convinced that $\vec{c}$ is well formed if the corresponding vector $\vec{e} = (e_1, \ldots, e_N)$ of plaintexts satisfies

$$\bigvee_{i=1}^{N} \big(e_i = 1 \ \wedge \ e_\ell = 0, \ \forall \ell \neq i\big).$$

Finally, the third condition is equivalent to verify that the value encrypted in the scalar product $\vec{b}_j \cdot \vec{c}$ is 1, for some $j \in \{1, \ldots, s\}$. In effect, assuming that $\vec{e}$ is well formed, the scalar product of the plaintext vectors, $\vec{b}_j \cdot \vec{e}$, will be 1 if and only if $i \in B_j$.

Putting all these pieces together, the t-th execution of this 'request and retrieval' protocol works as follows, for $t \geq 1$.

1) To retrieve item db_i, the client computes the vector $\vec{e} = (e_1, \ldots, e_N)$ such that $e_i = 1$ and $e_\ell = 0$, $\forall \ell \neq i$, and encrypts it coordinate-wise, to obtain $\vec{c} = (c_1, \ldots, c_N)$, where $c_\ell = \mathcal{E}_{pk}(e_\ell)$, for $\ell = 1, \ldots, N$.
2) The client sends $\vec{c}$ and $\mathcal{E}_{pk}(u)$ to the server, where $u = u_{t-1}$.
3) The server chooses at random u_t, and encrypts it. He also computes the value $\tilde{c} = \bigoplus_{\ell=1}^{N} (db_\ell) c_\ell$.
4) By using the protocols/ideas explained in Section II-B, the server performs a conditioned disclosure of the pair $(\mathcal{E}_{pk}(u_t), \tilde{c})$, under the condition

$$\left(\bigvee_{i=1}^{N} \left[e_i = 1 \ \wedge \bigwedge_{\ell=1, \ell \neq i}^{N} e_\ell = 0 \right] \right) \bigwedge \left(\bigvee_{j=1}^{s} \vec{b}_j \cdot \vec{e} = 1 \right) \bigwedge (u = u_{t-1}).$$

5) The server updates the value of the encryptions of the vectors $\vec{b}_j$, for $j = 1, \ldots, s$. Remember that the correct update is $\vec{b}_j := [\vec{b}_j \cdot \vec{e}](\vec{b}_j \ - \vec{e})$. Therefore, the server updates the ciphertexts

$$\vec{c}_j := [\vec{c}_j \odot \vec{c}](\vec{c}_j \ - \vec{c}),$$

for $j = 1, \ldots, s$, where $(c_1, \ldots, c_N) \odot (c'_1, \ldots, c'_N) = (c_1 \otimes c'_1) \oplus \ldots \oplus (c_N \otimes c'_N)$. He replaces the old values of $\vec{c}_j$ with the new ones, in $\texttt{info}_C$, where he also replaces u_{t-1} with u_t.
6) If the client has always behaved honestly, he can recover the pair $(\mathcal{E}_{pk}(u_t), \tilde{c})$ and decrypt both ciphertexts with sk, obtaining in this way u_t and the desired item db_i.

Because of the security properties of the encryption scheme and of the protocol for conditional disclosure of secrets, it is easy to see that this protocol enjoys the required privacy properties for restricted oblivious transfer. The server cannot obtain any information about the index i, and the client obtains item db_i only if he is allowed to do it.

VI. Conclusions

We have presented in this work the first solution to the general problem of restricted oblivious transfer, which works for any family of restrictions. Unfortunately, the proposed solution is currently of theoretical interest only; it cannot be implemented in practice, because it uses public key cryptosystems which are at the same time additively and multiplicatively homomorphic. Instances of such cryptosystems have not been found yet.

As future research related to this work, we can mention several possibilities. The first one would be to design a public key cryptosystem with both additive and multiplicative homomorphic properties, which would make our solution fully implementable. The second one would be to find a (completely) different way of solving the generic problem of restricted oblivious transfer, which circumvents the use of non-realized cryptographic primitives. Finally, we insist that proving or disproving our conjecture on the new characterization of weighted threshold families would have a significant impact.

References

[1] B. Aiello, Y. Ishai and O. Reingold. Priced oblivious transfer: how to sell digital goods. *Proceedings of Eurocrypt'01*, LNCS **2045**, Springer-Verlag, pp. 119–135 (2001).

[2] G.R. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference, American Federation of Information*, Processing Societies Proceedings **48**, pp. 313–317 (1979).
[3] D. Boneh, E-J. Goh and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *Proceedings of TCC'05*, LNCS **3378**, Springer-Verlag, pp.p 325–341 (1999).
[4] G. Brassard, C. Crépeau and J.M. Robert. All-or-nothing disclosure of secrets. *Proceedings of Crypto'86*, LNCS **263**, Springer, pp. 234–238 (1987).
[5] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, **31**, pp. 469–472 (1985).
[6] Y. Gertner, Y. Ishai, E. Kushilevitz and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, **60** (3), pp. 592–629 (2000).
[7] P. Morillo, C. Padró, G. Sáez and J.L. Villar. Weighted threshold secret sharing schemes. *Information Processing Letters*, .**70**, pp. 211–216 (1999).
[8] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Proceedings of Eurocrypt'99*, LNCS **1592**, Springer-Verlag, pp. 223–238 (1999).
[9] M. Rabin. How to exchange secrets by oblivious transfer. Technical report TR-81, Harvard Aiken Computation Laboratory (1981).
[10] A. Shamir. How to share a secret. *Communications of the ACM*, vol. **22**, pp. 612–613 (1979).